JPL contact: John Kelly 125-233

A pplying Formal Methods and Object-Oriented Design to Existing Flight Software

Betty Cheng
Michigan State University
Department of Computer Science
A714 Wells Hall
East Lansing, MI 48824-1029
chengb@cps .msu. edu

Brent Auernheimer
California State University, Fresno
Department of Computer Science
Fresno, CA 93740-0109
brent.auernheimer@CSUFresno.edu

Introduction

Correctness is paramount for safety- critical software control systeI Is. Critical software failures in medical radiation treatment [1], communications [2], and defense [3] are familiar to the public. The significant quaI itity of software malfunctions regularly reported to the software engineering community [4], the change in laws concerning liability [5], and a recent NRC Aeronautics and Space Engineering Board report [6] additionally motivate the use of error-reducing software development techniques.

The benefits of formal methods in requirements-driven software development ("forward engineering") is well documented [7, 8, 9, 10,]], 1 2]. One advantage of rigourously engi - neering software is that forI nal notations are precise, verifiable, and facilitate autoI nated processing [1 3].

We claim that maintenance of critical legacy code also benefits from formal methods. For example, formal specifications can be reverse engineered froI I existing code. TI Ie resulting formal specifications are then the basis for chaI ge requests and the foundation for sill.)scquellt, verification and validation. Considering re-implementation's high cost and, even worse, the failure of critical software, reverse engineering of code into forI nal specifications provides an alternative to traditional approaches for maintaining safety-critics] systems.

This paper describes a project appling formal methods to a portion of the shuttle on-orbit digital autopilot (DA I)). Three objectives of the project were to: deI monstrate the 11s[! of formal methods 011 a shuttle application, facilitate the incorporation and validation of new requirements for the system, and verify the safety-critical properties to be exhibited by the software.

IIi addition to developing formal specifications of a critical module, a graphical depiction of the subsystem was constructed using the *Object Modeling Technique* (OMT) to provide an object-oriented view of the system as it relates to the functional and dynamic views. I essons lea rned from this project are described, including disc ussions of the benefits of constructing and being able to generate proofs with the formal specifications.

Project Overview

A portion of the shuttle software was chosen for a formal methods demonstration project involving NASA's Jet Propulsion Laboratory, Johnson Space Center, and Langley Research Center [1 4]. A related project of a smaller scale was performed by the authors in conjunction with the larger demonstration project. The Phase_Plane module, the control system for automatic attitude control of the shuttle, was the subsystem selected for the smaller project. The criteria that led to the selection of Phase_Plane included finding a module with difficult to understand requirements and potential for critical change requests.

Three tasks were performed in the development of the formal specifications of the module's high-level requirements. First, an understanding of the original requirements was needed. This involved consulting the *Functional Subsystem Software Requirements* (FSSR) document (also known as Level C requirements, consisting largely of "wiring diagrams"), *Guidance and Control Systems Training Manual*, source code, informal design notes, and discussions with shuttle software personnel. An "as-built" formal specification capturing the functionality depicted by the FSSR "wiring diagrams" was then developed.

Second, when attempting to derive a more abstract requirements-level formal specification, it was difficult to eliminate the implementation bias present in the as-built layer. A level of OMT diagrams were developed to depict the information from the first level of specifications. These diagrams facilitated the abstraction process and lead to the next higher level of specifications. This iterative process consisting of developing a level of formal specifications, followed by constructing the corresponding OMT diagrams lead to identification of the high level, critical requirements of the Phase_Plane module. Example specifications and OMT diagrams are included in the full paper.

The third task involved outlining proofs between the levels of specifications developed. That is, each specification must be shown to correctly implement the more abstract specification above it. These proofs provide traceability from the implementation details as described by the "wiring diagrams" to the high level requirements.

References

[1] Nancy G. Leveson and Clark S. Turner. An investigation of the Therac-25 accidents. *IEEE Computer*, pages 18 41, July 1993.

[2] Leonard Lee. *The day the phones stopped.* New York: Donald I. Fine, 1991.

[3] Eric Schmitt. Army is blaming patriot's computer for failure to stop Dhahran scud. *New York Times*, May 1991.

[4] Peter G. Neumann. RISKS DIGEST. Internet mailing list and newsgroup, regular feature of ACM SIGSOFT *Software Engineering Notes.*

[5] Victoria Slid Flor. Ruling's dicta causes uproar. *The National Law Journal*, July 1 {~{]].

[6] Committee for the Review of Oversight Mechanisms for Space Shuttle Flight Software Processes, Aeronautics and Space Engineering Board, National Research Council. *An assessment of space shuttle flight software development processes.* Washington, D.C: National Academy Press, 1993.

[7] Jeannette M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8 '24, September 1990.

[8] Susan L. Gerhart. Applications of formal methods: Developing virtuoso software. *IEEE Software*, pages 7-] (), September 1990.

[9] Nancy G. Leveson. Formal methods in software engineering. *IEEE Transactions on Software Engineering*, 16(9):929 930, September **1990.**

[1 0] Richard A. Kemmerer. Integrating formal methods into the development process. *IEEE Software*, pages 37-50, September 1990.

[11] Anthony Hall. Seven myths of formal methods. *IEEE Software*, pages 1]] 9, September 1990.

[12] Dan Craigen, Susan Gerhart, and Ted Ralston. *An international survey of industrial applications of formal methods* (NIST GCR 93/626). National Institute of Standards and Technology, 1993.

[13] Betty H.C. Cheng. Synthesis of procedural abstractions from formal specifications. In *Proc. of COMPSAC'91*, pages 149 154, September 1991.

[14] JPL, JSC, LaRC, and IBM-Houston. Formal methods demonstration project for space applications.pop report FY 93). in preparation, 1993.